DOCUMENT RESUME

ED 285 585 IR 052 067

AUTHOR Lewin, D. R.
TITLE Interactive Graphics in CAD/CAI in Chemical Engineering.
PUB DATE Apr 86
NOTE 14p.; Paper presented at the IBM Academic Information Systems University AEP Conference (San Diego, CA, April 5-8, 1986). For the conference proceedings and agenda, see IR 051 770-771.
PUB TYPE Guides - Non-Classroom Use (055) -- Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Chemical Engineering; *Computer Assisted Instruction; *Computer Graphics; Computer Software; *Engineering Graphics; Foreign Countries; Higher Education; Man Machine Systems; *Microcomputers; Programing; Systems Development
IDENTIFIERS *Computer Assisted Design; *Israel; Mainframe Computers

ABSTRACT

This paper describes the development of a software program which incorporates interactive graphics techniques into a teaching and research environment at the Department of Chemical Engineering, Technion, Israel, and the experience of transferring the software from mainframe to personal computer (PC) operating systems at the California Institute of Technology (Caltech). Interactive graphics and interactive optimization are discussed as the two broad process design and analysis applications that require the incorporation of interactive procedures, and the development of a process control synthesis program (ProCoSP) is described. This program features two design domains and a performance domain to meet specific research and educational requirements. A comparison of the use of the program on mainframe and PC systems indicates that the added portability and convenience achieved by opting for a PC operating environment far outweigh the potential disadvantage of reduced computational power compared to mainframe systems. Finally, guidelines are presented for other educators proposing to develop scientific programming meant for users inexperienced in operating computers, and it is concluded that ProCoSP represents progress towards the development of software that will allow the novice user to manipulate all the relevant components of a program with the greatest convenience possible. Two tables and five figures supplement the text. (KM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

D.R. Lewin

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED285585

INTERACTIVE GRAPHICS IN CAD/CAI IN CHEMICAL ENGINEERING

Presenter: D. R. Lewin *California Institute of Technology*

copyright © California Institute of Technology, 1986

ABSTRACT

This paper will describe the developments of a new program incorporating interactive graphics techniques into a teaching and research environment at the Department of Chemical Engineering, Technion, Israel, and the experience of transfering this software from main-frame to PC operating systems at Caltech. Examples are given from studies in the fields of stability analysis, nonlinear dynamics and control synthesis. These illustrate the scope for interactive graphics as a means of liberating the student or designer from tedious numerical calculations, allowing him instead to concentrate on the application of his perception and creativity to the problem in hand.

The interactive process control synthesis program, ProCoSP, is introduced. This tool features two design domains (Root Locus and frequency domains) and the performance domain (Step response), and illustrates the advantages of interactive programming in process control synthesis. A further point to emphasize is the educational value of such a tool. By trying to design control for various process transfer functions using the program, a student of linear control theory armed with the theoretical background will strengthen his practical "feel" for the subject and gain insights into the relationships between the eigenvalue, frequency and time domains.

It was found that the added portability and convenience achieved by opting for a PC operating environment far outweighed the potential disadvantage of reduced computational power compared to main-frame systems.

IR052067

MOTIVATION

While the use of numerical simulation in process design and analysis is widespread, the use of the computer has in most cases been limited to batch-wise computation. However, very often the nature of certain tasks merits employing interactive procedures in which graphics output could be integrated. This statement could also be applied to Computer Aided Instruction (CAI),

where graphics display would in most cases aid the students understanding of the material being learned. Further, as observed by Himmelblau [4], interactive software also converts the user from passive observer to active participant in the learning process.

The two broad classes of applications which call for incorporating interactive procedures are:

Interactive Graphics

The use of interactive as opposed to batch graphics is of importance when an advantage can be gained by giving the graphics procedure control over the calculation of the displayed data. This would result from either a need to enlarge a section of the region under analysis in order to observe features of interest whilst preserving image resolution (see example in Figure 1) or when the next step in an analysis depends on the previous one (see example of the solution of an nonlinear ordinary differential equation in Figure 2).

Interactive "Optimization"

Interactive programming is justified in the solution of optimization problems when either a mathematical formulation of the constraints make a direct solution difficult, or when the relevant constraints cannot be determined a priori. These classes of problems are often characterized by multiple domains, where very often the design would be effected in some domains while the performance criteria are fixed in others. Iteration between the domains would then become a feature of the design.

Clearly, CAI employs features from both these classes, since the learning process is an iterative one. The goal our project at the Technion's Chemical Engineering faculty was to generate interactive graphics capability to meet this need. In addition to a basic interactive graphics library, an important project goal was the development of applications packages to meet specific research and educational requirements. The process control synthesis program described briefly here and in greater detail in a separate publication [5] was its first fruit.

Control synthesis is an iterative activity; design is effected in either the frequency or eigenvalue domain [2,6], but performance is defined in the time domain, and the translation to the design domains can only be approximated. For a first

pass, the program was limited to the design of the control structure brought in figure 3. Since stability of the controlled process depends on the process, G(s) and the control blocks C(s) and H(s), one can design compensators {C,H} which will stabalize the process in the face of disturbances, d. The feedforward block R(s) can then be included in order to add an extra degree of freedom in tuning for the desired tracking response (i.e. response of y to command r). Designing the control blocks involves manipulation of poles and zeros in order that the transfer function y/r obtained meets design specifications:

Offset–free tracking:
y –> r at time goes to infinity.

Overshoot limited by M:

$$\max_{t} \frac{y - r}{r} \leq M.$$

Response time less than T.

The program had therefore to allow for the natural iterative activity suggested above:

– Tuning the controllers CH(s) to design for a stable closed loop response (in the face of disturbances) in the chosen design domain, testing the result in the time domain and iterating to achieve the desired result.

– Factorization of the disturbance control, CH into two blocks, such that tracking response is optimized.

– Having selected the design of {C,H}, manipulation of the controller R(s) to improve the tracking response (seen in the time domain). Iteration to achieve the desired result.

The development of the program was spurred by the observation that much of the students' time was squandered in numerical computation either using cumbersome batch programs or worse, by hand. This meant that the true goal of the control design was seldom reached and that generally not enough experience was attained by the average student to make him confident of his proficiency in the material.

STRATEGY AND IMPLEMENTATION

The program was written in FORTRAN and organized as a series of modules linked by menus. To ease development, the programming task was subdivided into one main program and over sixty subroutines, making use of the linear systems library given by Melsa and Jones [7], and a Newton–Raphson iterative algorithm for solving the roots of a polynomial [9], as well as many subroutines especially written for this application.

This approach differs from that of Calo and Andres [1], whose APL software, whilst flexible, relies on user familiarity with both the program and his own options at each stage in the design; ProCoSP displays a menu of current options when relevant, thus guiding the novice. On the other hand, the program shares many features with the work of Edgar et al [3], but has fewer limitations. From industrial considerations, the authors justify only one class of plant transfer function. In addition, their program allows only one class of control structure (ProCoSP block C(s) only), and one class of controller algorithm (ideal PID). These constraints are not reasonable from an educational standpoint, and ProCoSP both frees the user to give any strictly proper transfer function for the plant, and allows for the more flexible control structure given in figure 3.

A logical flow diagram for ProCoSP is brought in figure 4. We note how little the modular structure of the program constrains the designer in iterating as required in order to arrive at his final design. In the first design step outlined above, the disturbance rejection design can be closed by the appropriate design of C and H. Having arrived at a design for these two blocks, the tracking response can be tuned by defining R(S) appropriately. This can be achieved by superimposing poles and zeros of R(s) on a map of closed loop poles and zeros derived from an expression involving G(s), H(s), and C(s). It is then possible to cancel undesired closed loop poles and zeros using zeros and poles in R(s).

The menu driven structure makes it possible to switch at will between the two design domains and the testing domain. In addition, from any module there are the additional options of:

– Hard copy of the screen display.

– Listing a summary of the design so far (i.e. of G(s), C(s), H(s), and R(s), together with step test performance (off–set, overshoot, rise time and settling time).

– Quitting the program.

A schematic showing the organization of the graphic display is shown in Figure 5. The user indicates his choice from the list of available alternatives by clicking the mouse at the selected option window. Numerical input, error messages and user instructions appear in the top band of the screen (Data Input Field), leaving the greater portion of the screen free for graphical display. In addition to driving the menus, the mouse is used universally as a locating device; the author has come to the conclusion that the mouse is the optimum input device, after experience with various alternatives (see Table 1).

The first versions of this program ran on the Technion's CDC Cyber 720 machine, under NOS2 (for Network Operating System, version 2). The primitive graphics operations employed CDC's TIGS library (for Terminal Independent Graphics System), which supports input from either keyboard, digitizer or touch sensitive screen. At Caltech, the program was transfered to PCs under DOS, utilizing MICROSOFT's Halo Graphics Library. It has successfully run on IBM PC XT's, IBM AT's, and Compaq personal computers, with IBM, Amdex and Hercules graphics cards. The minimum system requirements for running ProCoSP or utilizing the graphics library used to generate Figure 1 and 2 are listed in Table 2.

Whilst in principle, the main frame machine affords greater computational power, in practice the realities of a heavily loaded multi–user system result in no great advantage when compared to a PC. On the other hand, writing for PC application results in greater portability and cost–effectiveness compared to main frame.

CONCLUSIONS

To date, the main–frame version of ProCoSP has been used for two consecutive years at the Technion; this is not long enough to quantify the improvement in the average student. One can only gauge the success of the program from the expressions of approval brought by both students and educators. The most heartening responses were those offered by students who enjoyed the new learning experience. The decision to "download" the program to PC systems will go far in propagating its employment.

It should be emphasized that this project was developed by a Chemical Engineer with some programming experience; many hard lessons were learned from

experience. With this disclaimer in mind, the following guidelines are offered to fellow educators proposing to develope scientific programming meant for users inexperienced in operating computers.

– The instructions should be built into the program and be understood even by a "first–time" user. If relevant, include on–line "help".

– The user should be required to supply only essential information; the program should neither request "cosmetic" data nor that which is redundant.

– Most entry errors should be anticipated and the program should be designed to trap them and redirect the user as appropriate. Before releasing a program for general use, test it with appropriate "Guinea–pigs".

– Try never to leave the user staring at a blank screen (say when "number crunching") – plot results straight out onto screen as they are calculated or print a polite status message or both.

– Try to include in the program all the features you would have liked to have seen if you were in the user's place.

– Listen to user comments and take steps to modify your product to meet their requirements (when reasonable).

This program should be seen as one more step down the road to the perfect user–program interface. The ultimate goal is to generate software that will allow the novice user to manipulate all the relevant components of a program with the greatest convenience possible.

ACKNOWLEDGEMENT

Support of the Levi Eshkol (at the Technion), Chaim Weizmann and Fulbright (at Caltech) fellowships is gratefully acknowledged.

REFERENCES

[1] Calo, J. M. and Andres, R. P. "Use of Interactive Graphics–Based Software for Teaching Chemical Engineering Process Principles", Computers and Chemical Engineering, 5, 197 (1981)

[2] Coughanowr, D. A. And Koppel, L. b. "Process Systems Analysis and Control", McGraw–Hill (1967)

[3] Edgar, T. F., Heeb, R. and Hougen J. O. "Computer Aided Process Control System Design using Interactive Graphics", Computers and Chemical Engineering, 5, 225 (1981)

[4] Himmelblau, D. M., "Practical Experience in Using Graphics in Teaching Process Analysis and Simulation", Computers in Chemical Engineering, 5, 187 (1981)

[5] Lewin, D. R. "Interactive Graphics in CAD/CAI in Chemical Engineering", 2nd University AEP Conference, San Diego (1986)

[6] Luyben, W. L. "Process Modelling, Simulation and Control for Chemical Engineers", McGraw–Hill (1973)

[7] Melsa, J. L. and Jones, S. K. "Computer Programs for the Computational Assistance in the Study of Linear Control Theory", McGraw–Hill, 2nd, Ed. (1973)

[8] Pismen, L. To appear in AMS Symposium Series.

[9] "360 Scientific Subroutine Package, Programmers Manual", IBM No. H20 – 0205 – 3 (1986)

Table 1: Comparing alternative locator Devices to the mouse

Device	Objection
Cursor	Too slow
Touch Sensitive Screen	Resolution to coarse for consideration as a locator device.
Digitizer	Although resolution often better than that of screen, more expensive than mouse and also less portable.

Table 2: Minimum System Requirements

- IBM compatible PC's (The programs have successfully run on the IBM PC's, PC XT's and Compaq PC's).
- 256K memory.
- One floppy disk drive.
- IBM, Amdek or Hercules graphics cards.
- Serial port.
- MICROSOFT mouse.

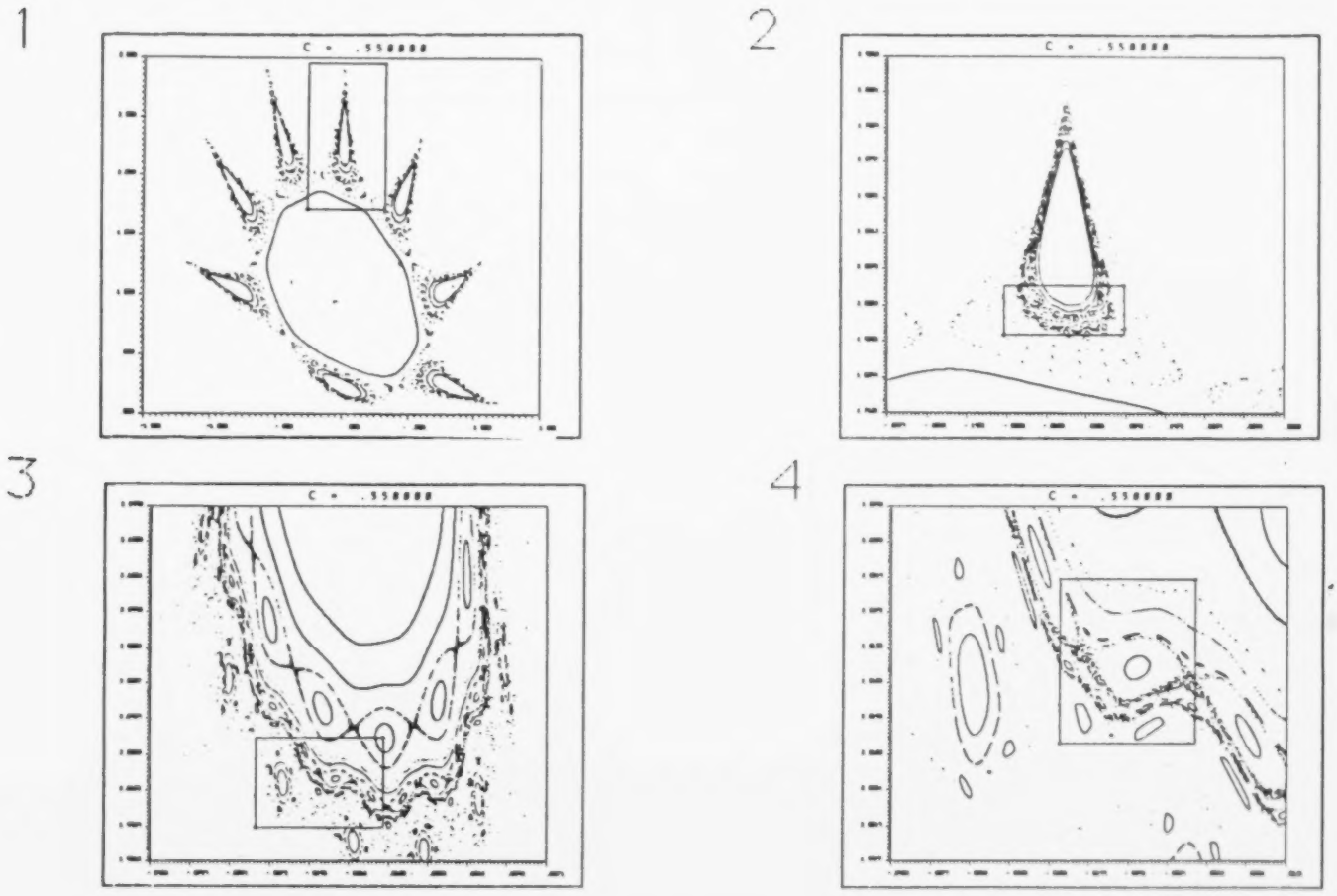

Figure 1.
Study of bifurcations from an approximation for the solution of differential equations representing coupled exothermic and endothermic kinetics (see Pismen [8]). The maps are generated by expressing the solution as an expansion in two coefficients, which are then plotted for a given value of a free parameter, labelled C. Each plot in the series is the rescaling of the window indicated in that preceding it.

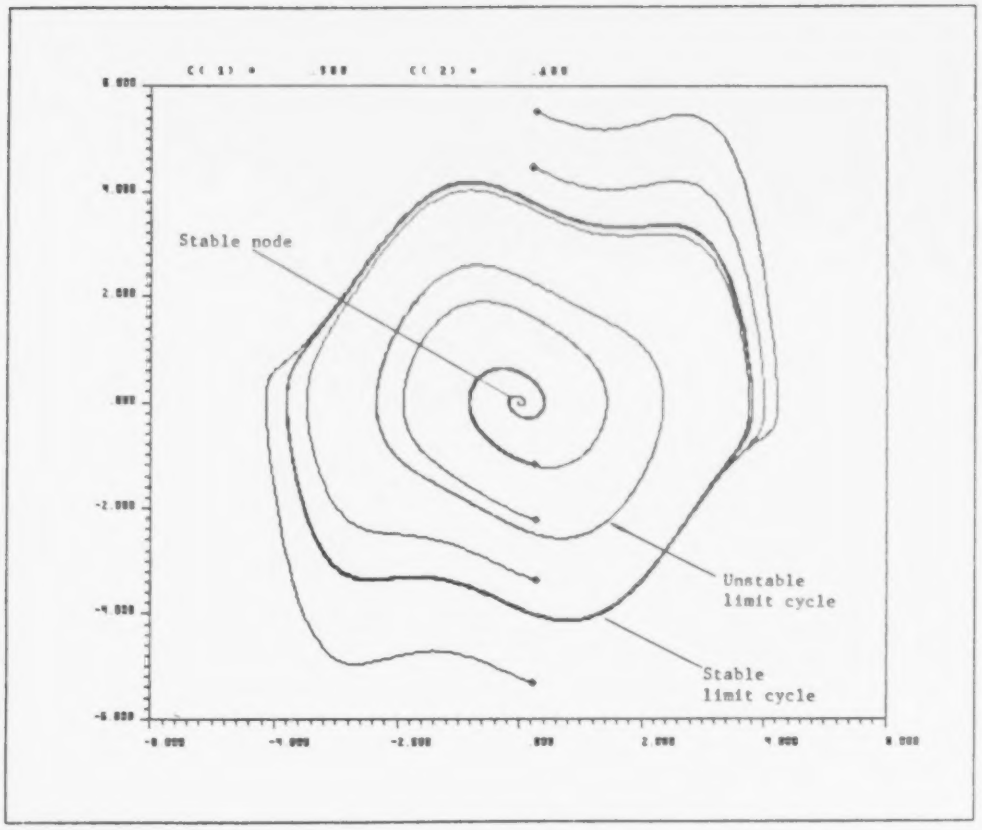

Figure 2. The solution in x–x space (phase diagram) of a Van der Pol's type equation:

$$\ddot{x} + c_1 \dot{x} (1 - x^2 + c_2 x^4) + x = 0$$

for various initial conditions and for given values of the constants c. In the phase diagram, the required initial conditions are entered using a mouse and the solution path appears following integration. In this way, the stable origin and the stable limit cycle can be located. Reverse polarity integration enables the identification of the unstable limit cycle.

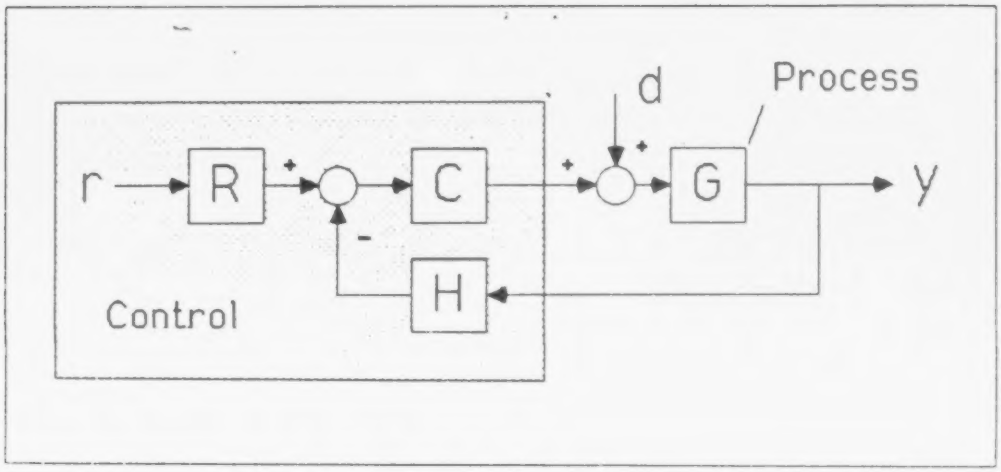

Figure 3. Control System Structure in ProCoSP.

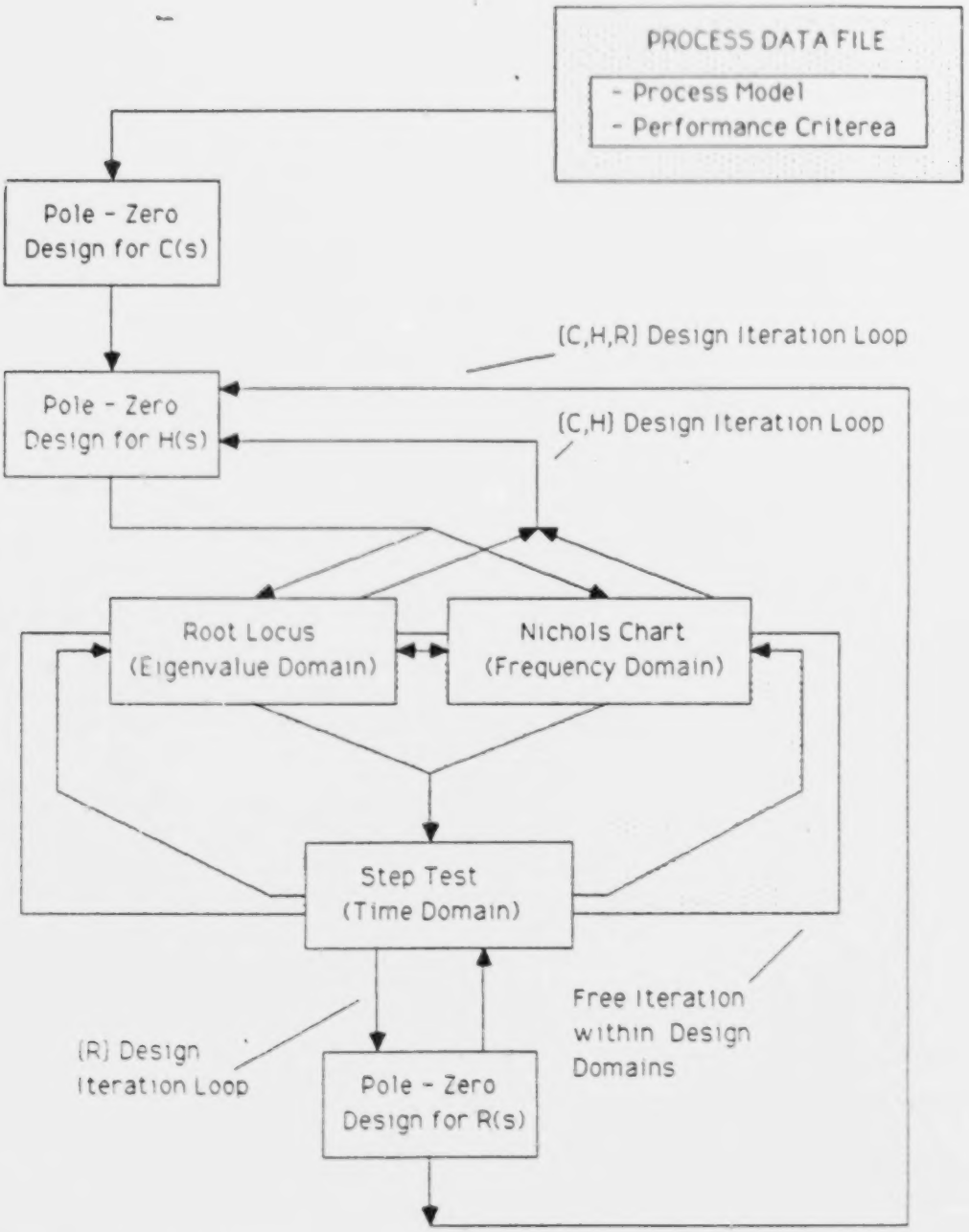
PROCESS DATA FILE
- Process Model
- Performance Criterea
Pole - Zero Design for C(s)
(C,H,R) Design Iteration Loop
Pole - Zero Design for H(s)
(C,H) Design Iteration Loop
Root Locus (Eigenvalue Domain)
Nichols Chart (Frequency Domain)
Step Test (Time Domain)
Free Iteration within Design Domains
(R) Design Iteration Loop
Pole - Zero Design for R(s)

Figure 4. Iteration Between Modules in ProCoSP.

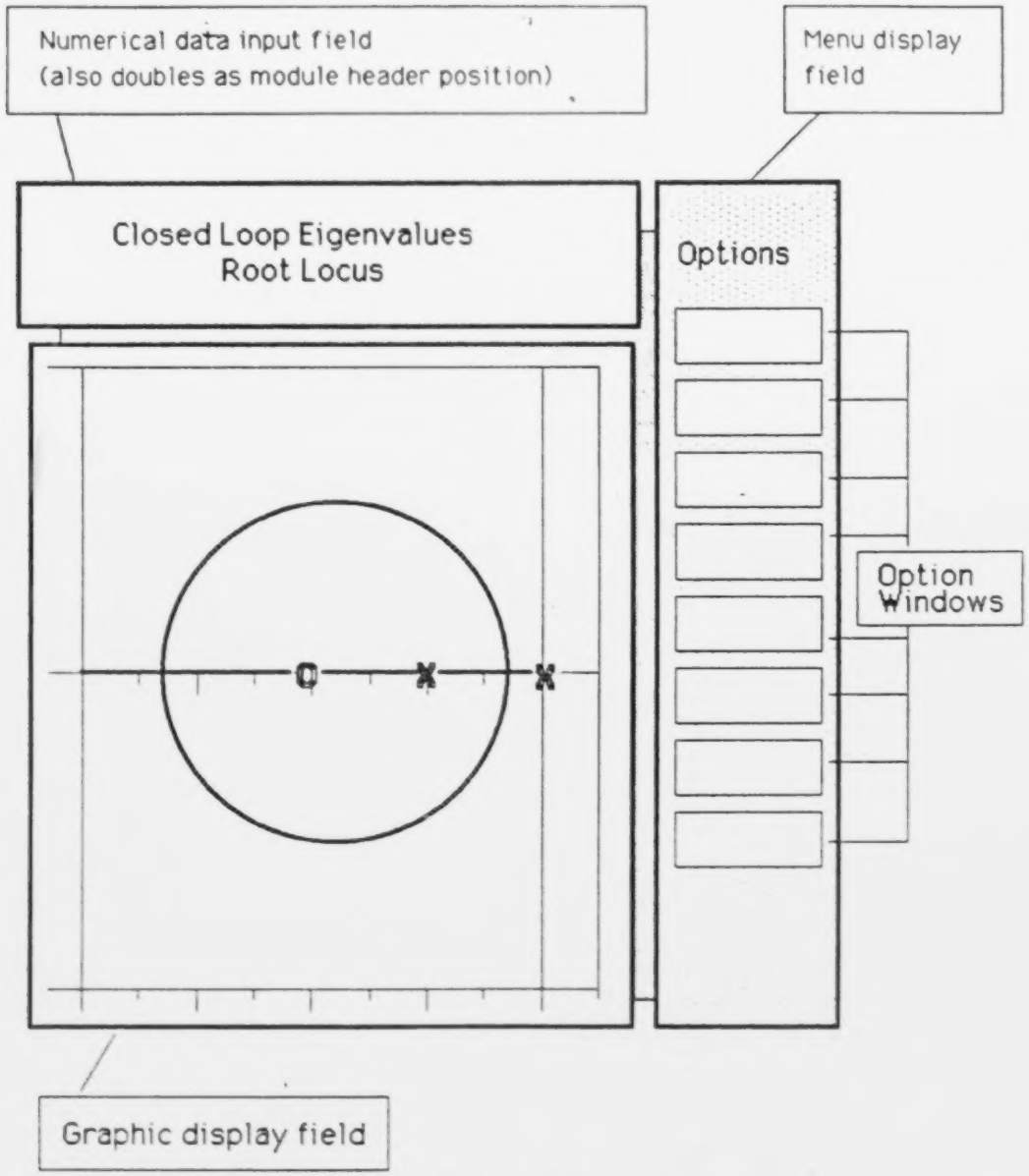

Figure 5. Organization of Screen Display in ProCoSP.